AN ENHANCER FOR REACTIVE PLANS

Diana F. Gordon
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Code 5514
Washington, D.C. 20375-5000

Abstract

This paper describes our method for improving the comprehensibility, accuracy, and generality of reactive plans. A reactive plan is a set of reactive rules. Our method involves two phases: (1) formulate explanations of execution traces, and then (2) generate new reactive rules from the explanations. Since the explanation phase has been previously described, the primary focus of this paper is the rule generation phase. This latter phase consists of taking a subset of the explanations and using these explanations to generate a set of new reactive rules to add to the original set. The particular subset of the explanations that is chosen yields rules that provide new domain knowledge for handling knowledge gaps in the original rule set. The original rule set, in a complimentary manner, provides expertise to fill the gaps where the domain knowledge provided by the new rules is incomplete.

1 INTRODUCTION

Reactive planning has proven to be a highly effective approach to planning (e.g., [8]). We have developed an enhancer for reactive plans that satisfies two goals. The first goal is to facilitate human understanding of plans generated by a particular class of reactive planners. This class consists of planners that gain effectiveness at the expense of human comprehensibility (e.g., [1] and [3]). The second goal is to improve the reliability and generality of reactive plans.

Our approach is divided into two phases: an *explanation phase* and a *rule generation phase*. The explanation phase begins with the application of a reactive planning system to a problem. The system generates an execution trace, which is translated into an abstract language trace. Next, a variant of Explanation Based Learning (EBL) is applied to explain the abstract trace using a previously constructed domain theory. The explanation process, which captures generalizations such as symmetries, culminates in English text describing the high level strategies of the reactive planner. We assume that each plan is composed of a set of reactive (i.e., stimulus-response) rules. During the rule generation phase, a subset of the explanations from the previous phase is used to generate a set of new reactive rules to add to the original plan. The new rules behave as a *local expert* because they remedy localized weaknesses of the original plan. The system may now use this enhanced plan to tackle the problem again. Our implementation of this method is called EXplain-and-GENerate (EXGEN).

The reactive planner's knowledge is imperfect. We know this because the planner, when using this knowledge (the original reactive plan), has less than a perfect success rate. Furthermore, the domain theory used to explain and generate rules is only complete for a local area of expertise. What is most interesting is that both sources of knowledge complement each other, one filling in the knowledge gaps for the other. When EXGEN is added to the reactive planner, the newly generated set of rules helps the planner to achieve 100% success for a problem described in Section 2.

Previous research is related to this work. EXGEN is similar to Learning Apprentice Systems (e.g., [2], [10]) because it uses EBL to explain the behavior of an expert and then uses this explanation to generate new planning rules. The difference is that Learning Apprentice Systems learn from a human to improve a system, whereas EXGEN learns from a system for the purposes of human understanding and system improvement. The type of EBL invoked by EXGEN is *Plausible EBL*, which has been developed by Gervasio and DeJong [2]. Our approach involves completing gaps in the domain theory for EBL. In terms of completing the gaps in a domain theory, several researchers provide solutions ([5], [10], [11], and [12]). However, we do not employ any of the previous approaches because their assumptions are not met in our situation. Instead, we use the original reactive plan to fill the knowledge gaps in the domain theory. Finally, the translation from high-level explanations to reactive rules is related to two techniques: specialization

[7] and symbol grounding [6].

Despite these similarities to previous research, the approach we present here is novel. Reactive planners are becoming increasingly complex in terms of their behavior. In response to this growing complexity, we have developed a plan enhancer. Ours is the first system that can explain the behavior of a reactive planner in human-oriented form, then improve the reactive plan based on the explanation. Since most people do not fully trust automation, especially if lives are at stake, people will probably want to screen system-generated plans for many applications. A plan enhancer can increase the acceptability of reactive plans to humans.

2 THE REACTIVE PLANNER AND DOMAIN

So far, we have tested our method with the SAMUEL system [4]. This system uses a genetic algorithm and other competition-based heuristics to learn high performance reactive plans in the absence of a strong domain theory. SAMUEL consists of three major components: a problem specific module, a performance module, and a learning module. In this research, we use a plan that has already been learned by the system. Therefore, our method only employs the problem specific and performance modules. Although reactive planning consists of both learning and executing a plan, for simplicity, we use the terms ''SAMUEL'' and ''reactive planner'' to refer to the performance module of this system.

SAMUEL has been applied to a variety of domains. We would like to see if EXGEN can improve this system's performance in each of these domains. Schultz and Grefenstette [9] have demonstrated that the addition of manually developed rules can improve SAMUEL's performance for the Evasive Maneuvers (EM) problem. Therefore, we have decided to begin testing our approach to automatic rule generation on this problem. In EM, which is simulated two-dimensionally, an agent (controlled by the reactive plan) tries to evade an adversary. The adversary tracks and tries to destroy the agent. An *episode* in EM begins with the adversary approaching the agent from a randomly chosen direction ends when either the agent is destroyed by the adversary or the adversary's speed falls below a given threshold. The latter occurs because, although the adversary's speed is initially greater than the agent's speed, the adversary loses speed as it chases the agent. Six sensors provide the agent with information about the current state: the agent's *last-turn*, the *time*, and the adversary's *range, bearing, heading,* and *speed*. There is one action variable to control the agent's *turning-rate* (also called *turn*).

An example of a rule learned by SAMUEL for EM is the following:

```
IF    (and (last-turn [-135..135]) (time [2..12])
           (range [0..700]) (bearing [2..6])
           (heading [0..30]) (speed [100..950]))
THEN  (turn 90)
```

where ''(S [X..Y])'' means that sensor S has values X through Y. Although individual rules are understandable, the general strategy underlying a chain of rule firings is not. Furthermore, these rules do not contain information about subgoals, such as ''increasing range of adversary'', that the rules can achieve.

3 EXPLANATION PHASE

EXGEN generates explanations of execution traces. An execution trace contains snapshots of sensor readings resulting from a sequence of actions. In EM, one trace corresponds to one episode, and the success or failure of the agent at evading the adversary is noted at the end of the trace. Currently, only *success traces* are explained (i.e., traces where the agent evades the adversary). EXGEN translates each trace to an abstract trace that contains qualitative terms in place of numeric values, then uses (Plausible) EBL to identify plausible *strategies* within the abstract trace. A strategy is equivalent to an EBL proof. A strategy consists of a set of *triggering preconditions* (i.e., sensor readings) and a *triggering action* (which may be a mathematical derivative of an action) followed by a chain of causal events which ultimately results in the satisfaction of a subgoal. A triggering action is directly controllable by the agent, whereas a subgoal is not. Finally, the system performs a largely cosmetic translation of each strategy to an English explanation. An example of an English explanation that EXGEN generated for EM is:

(E1) The triggering action of the agent, which is increasing turning rate, caused the adversary's turning rate to have value increasing, which caused the adversary's deceleration to have value increasing, which caused the subgoal, increasing adversary deceleration, to have been achieved between times 2 and 3. The triggering preconditions are: (1) the adversary's range is not far, and (2) the adversary's speed is high.

The domain theory for EBL is empirically derived by a program that calculates average frequencies of sensor and action values from previous success traces. A more detailed description of the explanation phase may be found in [3].

4 RULE GENERATION PHASE

The rule generation phase is the newest part of our method. This phase involves two activities: generating rules and selecting useful rules. The first activity is done by EXGEN, but the latter is a combined human-system effort.

EXGEN generates new reactive rules from explanations. Each rule is formed from the triggering preconditions and action of the strategy associated with an explanation. The rule is expected to achieve the subgoal for which it is a trigger. The mapping that is used to convert from abstract to concrete numeric values is the inverse of the mapping from concrete to abstract. An example rule that might be generated for the strategy associated with explanation E1 is:

```
(R1) IF (and (last-turn [45..45]) (range [0..500])
             (speed [400..1000]))
     THEN (turn 135)
```

This rule will increase the turning rate of the agent because the *last-turn* is 45 degrees and the next *turn* is 135 degrees.

A single explanation typically yields multiple rules. An important part of the rule generation phase is the selection of those new rules that can improve SAMUEL's performance. The reader should be aware during the remaining discussions that the way SAMUEL uses a reactive plan is highly complex due to partial rule matching and other features. Therefore, we do not have a clear idea of precisely how the new rules patch the gaps in the original rule set. Instead, we identify gaps in the original plan by analyzing planning failures that appear in the execution trace.

We have decided to add the newly generated rules to the original rule set, rather than to use them alone, because our domain theory is incomplete. Instead of consulting a domain expert to complete the theory, we adopt a complementary empirical-analytical approach to the problem. The original rule set, called ''rules-sam'', yields partial expertise and the new rules yield partial expertise.

How can this complementary blend of old and new rules be implemented? Experiments have indicated that an integration problem arises if this blend is performed arbitrarily. In other words, the new rules can force SAMUEL to enter parts of the search space for which no new rule applies and the rules from rules-sam do not provide adequate expertise. To illustrate with a specific EM scenario, suppose the actions taken at time T and ($T + 1$) using rules-sam are ''turn right'' and then ''turn left'' respectively. If the explanation process captures symmetry, then the action recommended by a new rule at time T might be ''turn left''. If the domain theory is incomplete, then perhaps no new rule would apply at ($T + 1$). If rules-sam also lacks knowledge for the new situation, then SAMUEL might apply the original rule that fired at ($T + 1$), which has the action ''turn left''. The pair of ''turn left'' actions at times T and ($T + 1$) would not perform the original direction reversal that was probably effective.

We have tried using genetic algorithms on the blended rule set to resolve the integration problem. Initial results are not encouraging. Therefore, instead of arbitrarily blending rules, we have chosen to decompose the main problem (e.g., EM) into two stages. A local expert is used to solve the latter stage. This is similar to the development of an expert to solve chess end games. The reason for developing an expert for the latter stage is that once control is transferred to this local expert, it will not return to rules-sam. Rule integration is therefore not a problem.

Let us illustrate how this method has been applied to EM. First, we have (manually) identified a latter stage of the EM problem. This required adding a new sensor *agent speed* to SAMUEL and running again on EM. Examination of failure traces reveals that rules-sam fails only when the adversary's speed is equal to the agent's speed. Furthermore, we know that once the adversary's speed goes below the agent's speed, the adversary's speed remains below it. The subproblem called *restricted EM*, in which the adversary's speed is less than or equal to the agent's speed, covers the knowledge gap of rules-sam and also satisfies the constraint that control need not be returned to rules-sam.

Next, EXGEN has developed a local expert, called ''rules-expert'', for restricted EM. Out of all the explanations generated by EXGEN, experiments have determined that two explanations suffice to generate this local expert. The first states that if the agent turns away from the adversary then the agent achieves the subgoal ''adversary behind agent'' (explanation E2). The second states that if the agent moves straight when the adversary is behind then the agent achieves the subgoal ''increasing adversary range'' (explanation E3). EXGEN converts the triggering preconditions and actions of these explanations into reactive rules. The rules that are derived from explanations E2 and E3 are mutually exclusive as well as exhaustive with respect to the ranges of values that they cover. This local expert rule set enables the planner to succeed 100% of the time over 1000 episodes of *restricted* EM.

To adapt these rules to be part of a plan for the original (*un*restricted) EM problem, EXGEN adds a test for ''adversary speed $\leq$ agent speed'' to the conditions of the new rules. Also, we have modified SAMUEL's conflict resolution mechanism to prefer the new EBL rules whenever they apply. Since the rules are mutually exclusive and exhaustive, this implies that once control moves to rules-expert, a deterministic sequence of actions are taken by the agent that never fail. These steps yield a local expert for restricted EM that fills the knowledge gap of rules-sam. The combination of rules-sam and this rules-expert is called ''rules-combo''.

The original plan, rules-sam, is a high-performing rule set. Using rules-sam, SAMUEL wins (i.e., evades the adversary) 992 out of 1000 episodes (99.2%). However, when the new rules are added to form rules-combo, the success over 1000 episodes climbs to 100% on the *original* EM problem. Although the performance improvement from 99.2% to 100% gained by adding the new rules to rules-sam would be insignificant for many applications, there are some applications for which this would be very meaningful. One example is the life-threatening situation of a human pilot using this plan to evade a missile that is tracking his plane. Furthermore, because they are based on generalized explanations (e.g., which capture symmetries), the new rules are more general than those in rules-sam and are therefore potentially applicable under more varied conditions.

5 SUMMARY AND FUTURE WORK

The goal of this research has been to improve the acceptability of reactive plans for human use. Two steps are taken toward this end, as described here: (1) human comprehensibility is improved, and (2) reliability and generality are increased. Using the approach that we present, it is possible to increase the trustworthiness of reactive plans.

Although comprehensibility is greatly improved by this method, the accuracy improves by only a small margin for the EM problem. This is due to the fact that the original plan developed by SAMUEL is already highly effective. In the future, we would like to apply this method to other problems on which SAMUEL is less effective to see if a much greater performance improvement can be achieved.

When applying EXGEN to other domains, we would also like to evaluate its generality. All of EXGEN's (implemented) learning methods seem likely to be applicable to a wide variety of domains. However, the separation of EM into stages to avoid integration problems may not be generally applicable. Future work will focus primarily on developing and implementing a more general method for handling rule integration.

Acknowledgements

I would like to thank John Grefenstette for suggesting the EXGEN project and providing SAMUEL. I would like to thank Bill Spears for his helpful comments, which include suggesting the use of a ''local expert''. I would also like to thank Jude Shavlik for indicating related literature, and members of the Machine Learning Group at NCARAI for proofreading this paper.

References

[1] Brooks, R. (1990). Elephants don't play chess. *Robotics and Autonomous Systems* 6: 3-15.

[2] Gervasio, M. and DeJong, G. (1989). Explanation-based learning of reactive operators. *Proceedings of the Sixth International Workshop on Machine Learning.* Morgan Kaufmann, Ithaca, NY.

[3] Gordon, D. and Grefenstette, J. (1990). Explanations of empirically derived reactive plans. *Proceedings of the Seventh International Conference on Machine Learning.* Morgan Kaufmann, Austin, TX.

[4] Grefenstette, J., Ramsey, C. and Schultz, A. (1990). Learning sequential decision rules using simulation models and competition. To appear in *Machine Learning.*

[5] Hall, R. (1986). Learning by failing to explain. *Proceedings of the Fifth National Conference on Artificial Intelligence.* Philadelphia, PA.

[6] Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42(1/3): 335-346.

[7] Langley, P. (1987). A general theory of discrimination learning. *Production System Models of Learning and Development.* D. Klahr, P. Langley, and R. Neches (eds), The MIT Press, Cambridge, MA.

[8] Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* Milan, Italy.

[9] Schultz, A. and Grefenstette, J. (1990). Improving tactical plans with genetic algorithms. *Proceedings of the Tools for Artificial Intelligence Conference.* Herndon, VA.

[10] Tecuci, G. and Kodratoff, Y. (1990). Apprenticeship learning in imperfect domain theories. *Machine Learning: An Artificial Intelligence Approach, Volume III.* Y. Kodratoff and R. Michalski (eds), Morgan Kaufmann.

[11] Van Lehn, K. (1990). Learning one subprocedure per lesson. *Artificial Intelligence,* 31(1): 1-40.

[12] Wilkins, D. (1990). Knowledge base refinement as improving an incorrect and incomplete domain theory. *Machine Learning: An Artificial Intelligence Approach, Volume III.* Y. Kodratoff and R. Michalski (eds), Morgan Kaufmann.